PATENT
Docket No. **VIV/0011.01**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Gregor P. Freund	Examiner: Ha, Leynna A
Serial No.: 10/605,189	Art Unit: 2188
Filed: September 12, 2003	APPEAL BRIEF
For: Security System with Methodology for Interprocess Communication Control	

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

BRIEF ON BEHALF OF GREGOR P. FREUND

This is an appeal from the Final Rejection mailed August 19, 2008, in which the currently-pending claims stand finally rejected. Appellant filed a Notice of Appeal on November 24, 2008. This brief is submitted electronically in support of Appellant's appeal.

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The real party in interest is assignee Check Point Software Technologies, Inc. located at 800 Bridge Parkway, Redwood City, CA 94065.

2. RELATED APPEALS AND INTERFERENCES

There are no appeals or interferences known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

The status of all claims in the proceeding is as follows:

Rejected: Claims 1-47

Allowed or Confirmed: None

Withdrawn: None

Objected to: None

Canceled: None

Identification of claims that are being appealed: Claims 1-47

An appendix setting forth the claims involved in the appeal is included as the last section of this brief.

4. STATUS OF AMENDMENTS

Two Amendments have been filed in this case. Appellant mailed a first Amendment on January 16, 2007, in response to a non-final Office Action dated October 16, 2006. Appellant mailed a second Amendment on April 23, 2008, in response to a non-final Office Action dated January 23, 2008. In response to the latter, the Examiner issued a Final Rejection dated August 19, 2008. Appellant filed a Notice of Appeal on November 24, 2008. In the Amendments, the pending claims were amended in a manner which Appellant believes clearly distinguished the claimed invention over the art of record, for overcoming the art rejections. Thus, Appellant has chosen to forgo filing an Amendment After Final which might further limit Appellant's claims, as it is believed that further amendments to the claims are not warranted in view of the art. Accordingly, no Amendments have been entered in this case after the date of the above-referenced

Final Rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

Appellant asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 1**: in a computer system operating under control of an operating system supporting interprocess communication, a method for controlling interprocess communication occurring between an application executing on the computer system and a service provided by the operating system (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: defining rules indicating which system services of the operating system a given application can invoke using interprocess communication to invoke said system services (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); trapping an attempt by a particular application to invoke a particular system service (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying the particular application that is attempting to invoke the particular system service (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); and based on identity of the particular application and on the rules indicating which system services a given application can invoke, blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

Appellant further asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 14**:in a computer system

operating under control of an operating system supporting interprocess communication, a method for regulating communications between processes that attempt to use said interprocess communication (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: defining a policy specifying whether one process may use interprocess communication of the operating system to communicate with another process (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); intercepting an attempt by a first process to communicate with a second process (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying the first process that is attempting to communicate with the second process [and] identifying the second process (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); based on said policy, determining whether the first process may communicate with the second process (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)); and allowing the first process to communicate with the second process if said policy indicates that the first process may communicate with the second process (see, e.g., Appellant's specification at aforementioned [0078]; also refer to [0071] where IPCC's allowing and blocking of messages is described).

Appellant further asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 25**: in a computer system operating under control of an operating system supporting interprocess communication, a method for controlling interprocess communications from one application to another (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: registering a first application to be protected from interprocess communications of other applications (see, e.g.,

Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); detecting an attempt to access the first application using interprocess communication (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying a second application that is attempting to access the first application using interprocess communication (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); and rerouting the attempt to access the first application through an interprocess communication controller that determines whether to allow the attempt, based on rules indicating whether the second application may access the first application using interprocess communication (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

Appellant further asserts that the art rejections herein fail to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 36**: a system for regulating interprocess communication between applications (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the system comprising: a computer having at least one processor (see, e.g., Appellant's specification at 100 (Fig. 1B) and paragraphs [0044-0051]), said computer operating under control of an operating system providing interprocess communication (see, e.g., Appellant's specification at paragraphs [0052-0055]); a policy specifying applications that are permitted to communicate with a first application using interprocess communication (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); a module for detecting a second application attempting to communicate with the first application using interprocess communication (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); and an

interprocess communication controller for identifying the second application attempting to communicate with the first application and determining whether to permit the communication based upon the identification of the second application and the policy specifying applications permitted to communicate with the first application (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection; see also, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

6. GROUNDS OF REJECTION TO BE REVIEWED

The grounds presented on appeal are:

(1st) Whether claims 1-47 are unpatentable under 35 U.S.C. 102(b) as being anticipated by Ablay et al. (USP 6,002,941, "Ablay") or, in the alternative, under 35 U.S.C. 103(a) as being obvious over Johnson et al. (USP 5,133,053, "Johnson").

A. First Ground: Claims 1-47 rejected under Section 102 or, alternatively, under Section 103

1. General

Under Section 102, a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in the single prior art reference. (See, e.g., MPEP Section 2131.) Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under Section 103, the Examiner must establish: (1) that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a

reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). As will be shown below, the art of record fails to teach each and every element set forth in Appellant's independent claims, as well as other claims, and therefore fails to establish anticipation of the claimed invention under Section 102, or obviousness under Section 103.

2. Claims 1-47 rejected on the basis of Ablay

Claims 1-47 stand rejected under 35 U.S.C. 102(b) as being anticipated by Ablay, et al. (US 6,002,941). The Examiner's rejection of claim 1 is representative:

> Ablay discloses in a computer system operating under control of an operating system supporting interprocess communication, a method for controlling interprocess communication occurring between an application executing on the computer system and a service provided by the operating system, the method comprising:
>
> defining rules (col. 8, lines 53-67) indicating which system services of the operating system (col. 4, lines 15-23 and col. 5, lines 31-37) a given application can invoke using interprocess communication to invoke said system services; (col. 2, line 65 - col. 3, lines 9 and col. 6, lines 5-9)
>
> trapping an attempt by a particular application to invoke a particular system service; (col. 3, lines 13-22 and col. 4, lines 34-52)
>
> identifying the particular application that is attempting to invoke the particular system service; and (col. 5, lines 62-67 and col. 12, lines 48-60)
>
> based on identity of the particular application and on the rules indicating which system services a given application can invoke (col. 4, lines 56-65 and col. 8, lines 53-67), blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service. (col. 3, lines 1-9 and col. 10, lines 5-18).

For the reasons set forth below, Appellant's claimed invention may be distinguished on a variety of grounds.

In the discussion that follows, Appellant will present technical information in sufficient detail to allow the Office to appreciate core architectural, design, and operational features that distinguish Appellant's invention from the prior art. The information provides (among other things) background and context to assist the reader with understanding the technology underlying the present invention, and there is no intention to suggest that "Appellant is arguing" each and every feature or point of discussion in the section that follows is present in Appellant's patent claims, as such would impractical, unwise, and not necessitated by the art of record. Instead, the following section provides a foundation upon which one may understand the problems presented by prior art approaches and the solution provided by the present invention. It is followed by specific discussion of claimed features that are believed to distinguish Appellant's claimed invention from the prior art.

Appellant's invention and patent claims are directed to controlling access of potentially "bad" applications or processes, such as "malware" inadvertently downloaded from the Internet, that may surreptitiously invoke operating system services (e.g., Microsoft Windows DNS service) using interprocess communication (IPC) technique for compromising a user's computer. Prior art computer systems have classically adopted user-based approaches to controlling security -- that is, by assuming that if "bad" (i.e., unauthorized) users are denied access to operating programs then computer system security is achieved. Malware attacks however do not involve "bad" users gaining access but, instead, involve "good" (i.e., authorized) users accidentally executing "bad" programs which in turn attack the computer system (e.g., for gaining access to sensitive information). For example, a "good" user may be duped into executing a malware program by clicking on (launching) an e-mail attachment. More recently, computer systems have attempted to improve security beyond simple user-based approaches by identifying and blocking the "bad" programs (i.e., malware) themselves. Hackers and malware purveyors have not sit still, however, but instead have taken additional steps to effectively cloak malware from such detection. Malware attacks using IPC are

particularly insidious to detect, because the malware (i.e., "bad") program hides its activity by running it through what is normally a safe and "good" operating system service or other "good" program (i.e., previously established to not be malware). Prior art program-based security systems do not detect or prevent this type of attack, which relies on IPC invocation of "good" (friendly) programs or processes, especially operating system services.

In order to understand Appellant's claimed invention, it is first necessary to understand what the term "interprocess communication" actually means, as that term is commonly used in the software-related art. This is particularly true given the latest rejection, where the Examiner cites a wireless communication system that has little or nothing to do with interprocess communication (it does not even mention the term) and, Appellant believes, is likewise not applicable to Appellant's claimed system and methodology for controlling interprocess communication with system services.

Microsoft's Computer Dictionary defines "interprocess communication" as follows:

> **Interprocess communication**: Abbreviated IPC. The ability, provided by a multitasking operating system, of one task or process to exchange data with another. Common IPC methods include pipes, semaphores, shared memory, queues, signals, and mailboxes.
> (*Microsoft Press, Redmond, WA, 1991)*

IBM's Dictionary of Computing defines the term along similar lines:

> **interprocess communication** (1) In the OS/2 operating system, the exchange of information between processes or threads through semaphores, queues, and shared memory. (2) In the AIX operating system, the process by which programs communicate data to each other and to synchronize their activities. Semaphores, signals, and internal message queues are common methods of inter-process communication. (3) In AIX Enhanced X-Windows, a communication path.

(International Business Machines Corporation, Research Triangle Park, NC, 1994)

Further (and detailed) discussion of interprocess communication, as it is known and practiced in the software art, is provided by the well-known Microsoft Developer Network (MSDN). (See, e.g., *MSDN Library > Win32 and COM Development > System Services > Interprocess Communications* at MSDN.Microsoft.com) Needless to say, the term is understood in the software-related art to refer to a particular type of communication or exchange of data that occurs between processes. When used in such a proper manner, the term is not haphazardly applied to all sorts of disparate ideas or technologies, such as now attempted by the Examiner.

Similarly, Appellant used the term "service" as it is understood in the software-related art to refer to a special type of application that serves other applications by providing special functionality or performing particular tasks on behalf of the other application. This is made explicit in the Glossary section of Appellant's Application:

> **Service**: A service is a special type of application that serves other applications by providing special functionality or performing particular tasks on behalf of the other application. Most modern operating systems typically include subsystems which make available a number of different services to applications which are running on the operating system. In the Microsoft Windows environment, for example, the operating system provides a LPC (local procedure call) facility which enables efficient communication with subsystems providing services. An example of a service is a DNS service provided by a subsystem of the Windows operating system subsystem that performs DNS lookup services on behalf of another application. For further information regarding services, see e.g., Dabak, Prasad et al, "Undocumented Win NT", October 1989, M&T Books, the disclosure of which is hereby incorporated by reference. An on-line copy of this book is currently available via the Internet at www.windowsitlibrary.com.

When used in such a proper manner (e.g., Appellant's claim 1: "a service provided by the operating system"), the term refers a particular type of feature available in modern-day operating systems. The term is not used casually in Appellant's application or claims to refer to any sort of vague notion of "service", such as phone service, wireless service, or room service, and certainly not Ablay's "emergency call", "private conversation", or "temporary talk group" wireless services.

In Appellant's application and claims, Appellant used the terms interprocess communication (IPC) and (operating system) service in a manner consistent with the foregoing definitions, to describe a very specific approach to controlling communications that may occur internally within computer systems. The specific vulnerability addressed by Appellant's invention involves the scenario where a malicious process (malware) has been placed on the user's machine. Malicious applications may seek to exploit well-known services that exist on a system, such as those that are published by services components or by the operating system. In present-day operating systems, such as Microsoft® Windows for example, a lot of interaction occurs between different processes (e.g., applications, drivers, and the like) which are running on a computer system at the same time. Moreover, considerable interaction occurs between different applications and these so-called "services" -- the special case applications that exist to serve other applications (e.g., by providing special functionality or performing particular tasks).

Consider, for example, the DNS (Domain Name System) service provided by Microsoft Windows XP. The DNS "resolver" service in Windows XP performs DNS lookup on behalf of other applications. DNS is itself normally a harmless protocol that contacts a DNS server for translating domain names (e.g., cnn.com) into IP addresses. However, a malicious application has the ability to use Windows' built-in DNS service to communicate with a malicious DNS server. For example, the malicious application may use the DNS service for a DNS lookup of "MySecret.Hacker.com". The DNS server at the hacker site ("Hacker.com") would then get a query from the local DNS server asking whether it has an IP address for "MySecret". In fact, what the hacker site DNS server receives is a token (string of "MySecret"). At this point, the malicious application may engage in almost unlimited communication with the malicious DNS server using an

awkward, but also very effective, protocol. During operation of a user's computer, a service component (such as the above-illustrated DNS resolver) may create a "subscription" address or "port" that a potentially malicious application may use to talk to the service. The created subscription address or port serves as a mechanism that allows a potentially malicious application to indirectly communicate with other applications and resources (e.g., with a malicious server) using specifically formatted messages. Therefore, in accordance with Appellant's invention, an application's ability to engage in such communication is regulated for the purpose of enforcing security policies.

During prosecution, Appellant's claims were amended to highlight the specific features of Appellant's invention that address the above vulnerability that is posed by this particular type of communication that occurs within a computer, when one application or process attempts to use interprocess communication to invoke operating system services in a manner that thwarts security measures. Importantly, security is not based on the identity of a particular user or his/her role or authorization, or even on the identification of a particular "bad" program, but is instead based on the notion that certain operating system services or other "good" processes are vulnerable to invocation by malicious programs, for example allowing the operating system to be tricked into carrying out the malicious or compromising act. Vulnerable operating system services in particular should be protected from IPC invocation by malicious programs, regardless of how safe or authorized the currently logged-in user is thought to be.

In response to Appellant's amendments, the Examiner has revamped the rejection by attempting to map Appellant's claim limitations into unrelated art. In particular, the Examiner now intends to map Appellant's claim limitations into a wireless communication system that has nothing to do with interprocess communication and nothing to do with operating system services (such as ones akin to the DNS service). For example, Ablay describes his "services" as pertaining to "services to wireless units (e.g., mobile and/or portable radios)." (Ablay Col. 1, lines 14-15). Examples provided by Ablay include "temporary talk groups" and "other services akin to the creation of temporary talk groups are also commonly available, such as emergency alarm, emergency call, private conversation, and call alert." A service such as "call alert" is obviously a high level or user-perceptible wireless service, and bears little or no

resemblance to low-level operating system services (e.g., operating system DNS service) which are not perceptible to the user. Similarly, Appellant's claim limitation of "interprocess communication" has no realistic mapping to anything in Ablay. For example, the Examiner points to Ablay at Col. 2, Line 65 - Col. 3, Line 9, and Col. 6, Lines 5-9, however, a detailed review of those sections reveals no discussion whatsoever of anything remotely related to the general notion of interprocess communication (as that term is properly understood in the art).. In fact, there is no discussion at all of any sort of communication or exchange of data among processes concurrently running on a given device (computer or otherwise). Instead, Ablay gives a rather general discussion of invocation and termination of services.

Ablay's wireless service approach has little in common with Appellant's invention. Significantly, if Ablay's system were inadvertently infected with malware by a user (e.g., an authorized user being duped into downloading malware), there simply is no facility or means described for Ablay's system to prevent the malware from executing and potentially compromising the system (e.g., gaining access to sensitive data or even gaining control over the entire system). In fact, Ablay is entirely silent regarding any malware or security-related concerns, and a search of the patent fails to turn up even a single reference to "malware", "virus", "Trojan horse", or even "security." Unlike the Examiner's previous rejection (based on Andrews), Ablay is not even a security-related patent. Simply put, Ablay is far removed from Appellant's invention, or even field of technology. Ablay does not contain the same features and does not function in the same way as Appellant's invention, and clearly Ablay cannot achieve the same result.

All told, Ablay's teaching is directed to activating wireless services, such as "caller alert." Ablay has no description or notion that a rogue program may attempt to invoke another (innocent) process to masquerade on its behalf (such as, duping the operating system to post sensitive information to a hacker's DNS server). Ablay does not even have any discussion about a rogue program being accidentally or inadvertently executed by a user, authorized or not. Appellant's claims set forth a patentable advance in the area of controlling access of potentially "bad" applications that may cloak their malicious activities through interprocess communication. Appellant's claims were amended during prosecution to highlight the specific features of Appellant's invention

that address the vulnerability posed by interprocess communication, that one application or process may attempt to use interprocess communication to thwart security measures by running its malicious activities through what is considered to be a "good" (non-malware) process or service, thereby hiding its activities from prior art security measures (e.g., prior art firewalls). Accordingly, it is respectfully submitted that the claims distinguish over Ablay and that the Examiner's rejection under Section 102 should not be sustained.

3. Claims 1-47 alternatively rejected under Section 103 as obvious over Johnson

Claims 1-47 also stand rejected, in the alternative, under 35 U.S.C. 103(a) as "obvious over Johnson et al." (5,133,053, "Johnson"). Here, the Examiner states on page 5 of the Final Rejection that Johnson is "brought forth" for the proposition that "interprocess communication" is well-known. As Appellant already stated in his Background Section and throughout his Application, "interprocess communication" per se is well-known.

In bringing forth Johnson, Examiner refers back to Ablay (e.g., at page 8 of the Final Rejection). Therefore, the alternate rejection as currently written is structured as "obvious over Ablay in view of Johnson." (If the Examiner believes the Section 103 rejection is based on Johnson alone, it is respectfully requested that the Examiner withdrawal the Final Rejection and reissue it in proper form, striking those portions of the "Johnson rejection" that reference and rely on Ablay.) Appellant believes the claims to be allowable over Ablay in view of Johnson for least the reasons stated above pertaining to the Section 102 rejection based on Ablay. The Ablay patent does not pertain to security-related applications, nor does Ablay discuss operating system-level services or how communication with them (via IPC or any other means) can be controlled to thwart malicious software. Johnson does not cure any of these deficiencies. The claims are also believed to be allowable for least the following additional reasons.

On page 5 of the Final Rejection, the Examiner's statement regarding Johnson, as best as understood, is that the fact that interprocess communication may occur in the operating system of a computer is known (i.e., prior art). As Appellant has stressed above and throughout his Application, interprocess communication is well known and occurs when one task or process to exchange data with another, whether they be between

two application processes, and application process and an operating system process, or to operating system processes. Johnson shows, if anything, that the Examiner is not at liberty to use the term interprocess communication loosely to cut across a wide swath of potential data exchanges (as the Examiner has done in mapping Appellant's claims to Ablay). As far as Johnson's teaching of interprocess communication itself, he discloses a mechanism for facilitating communication between processes across different nodes in a network; there, however, is no disclosure pertaining to identifying and blocking service requests that occur via interprocess communication. As such, Johnson adds nothing beyond the IPC concepts and prior art already disclosed in Appellant's Background Section.

For the reasons stated, it is respectfully submitted that the references when combined, or taken individually, do not teach or suggest Appellant's claimed invention of controlling access to system services via IPC. Appellant's approach effectively thwarts "bad" applications or processes, such as "malware" inadvertently downloaded from the Internet, from invoking operating system services (e.g., Microsoft Windows DNS service) using interprocess communication (IPC) technique. Both prior art references are entirely silent with respect to such features (or even the underlying problem being addressed). Accordingly, it is respectfully submitted that the claims distinguish over the references and the Examiner's rejection under Section 103 should not be sustained.

B. Conclusion

The present invention greatly improves the ease and efficiency of the task of controlling access of potentially "bad" applications or processes to sensitive operating system services. It is respectfully submitted that the present invention, as set forth in the pending claims, sets forth a patentable advance over the art.

In view of the above, it is respectfully submitted that the Examiner's rejections under 35 U.S.C. Section 102 and 103 should not be sustained. If needed, Appellant's undersigned attorney can be reached at 408 884 1507. For the fee due for this Appeal Brief, please refer to the attached Fee Transmittal Sheet. This Brief is submitted electronically.

Respectfully submitted,

Date: February 24, 2009

/John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX

7. CLAIMS APPENDIX

1. In a computer system operating under control of an operating system supporting interprocess communication, a method for controlling interprocess communication occurring between an application executing on the computer system and a service provided by the operating system, the method comprising:

defining rules indicating which system services of the operating system a given application can invoke using interprocess communication to invoke said system services;

trapping an attempt by a particular application to invoke a particular system service;

identifying the particular application that is attempting to invoke the particular system service; and

based on identity of the particular application and on the rules indicating which system services a given application can invoke, blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service.

2. The method of claim 1, wherein said trapping step includes intercepting operating system calls for invoking the particular system service.

3. The method of claim 1, wherein said trapping step includes intercepting local procedure calls for invoking the particular system service.

4. The method of claim 1, wherein said trapping step includes intercepting an attempt to open a communication channel to the particular system service.

5. The method of claim 1, wherein said trapping step includes rerouting an attempt to invoke the particular system service from a system dispatch table to an interprocess communication controller for determining whether to block the attempt based on the rules.

6. The method of claim 5, wherein said step of rerouting attempts to invoke the particular system service from a dispatch table to the interprocess communication

controller includes replacing an original destination address in the system dispatch table with an address of the interprocess communication controller.

7. The method of claim 6, further comprising the steps of:

retaining the original destination address; and

using the original destination address for invoking the particular system service if the interprocess communication controller determines not to block the attempt.

8. The method of claim 1, wherein the rules specifying which system services a given application can invoke are established based on user input.

9. The method of claim 1, wherein the step of blocking the attempt is based upon consulting a rules engine for determining whether the particular application can invoke the particular system service.

10. The method of claim 1, wherein the step of blocking the attempt includes obtaining user input as to whether the particular application can invoke the particular system service.

11. The method of claim 10, wherein said step of obtaining user input as to whether the particular application can invoke the particular system service includes the substeps of:

providing information to the user about the particular application that is attempting to invoke the particular system service; and

receiving user input as to whether the particular application should be blocked from invoking the particular system service.

12. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

13. The method of claim 1, further comprising:

downloading a set of computer-executable instructions for performing the method of claim 1.

14. In a computer system operating under control of an operating system supporting interprocess communication, a method for regulating communications between processes that attempt to use said interprocess communication, the method comprising:

defining a policy specifying whether one process may use interprocess communication of the operating system to communicate with another process;

intercepting an attempt by a first process to communicate with a second process;

identifying the first process that is attempting to communicate with the second process;

identifying the second process;

based on said policy, determining whether the first process may communicate with the second process; and

allowing the first process to communicate with the second process if said policy indicates that the first process may communicate with the second process.

15. The method of claim 14, wherein the first process comprises an instance of an application program.

16. The method of claim 14, wherein the second process comprises a system service.

17. The method of claim 14, wherein said intercepting step includes intercepting operating system calls made by the first process to attempt to communicate with the second process.

18. The method of claim 14, wherein said intercepting step includes detecting local procedure calls.

19. The method of claim 14, wherein said intercepting step includes detecting an attempt by the first process to open a communication channel to the second process.

20. The method of claim 14, wherein said intercepting step includes rerouting attempts by the first process to communicate with the second process from a system dispatch table to an interprocess communication controller.

21. The method of claim 14, wherein said step of identifying the second process includes evaluating parameters of the attempt made by the first process to communicate with the second process.

22. The method of claim 14, wherein said policy specifies particular processes to be protected from communications made by other processes.

23. The method of claim 14, further comprising:

providing for a process to be registered in order to be protected from communications made by other processes; and

determining whether to allow the first process to communicate with the second process based, at least in part, upon determining whether the second process is registered.

24. The method of claim 23, wherein said determining step is based, at least in part, on the type of communication the first process is attempting with the second process.

25. In a computer system operating under control of an operating system supporting interprocess communication, a method for controlling interprocess communications from one application to another, the method comprising:

registering a first application to be protected from interprocess communications of other applications;

detecting an attempt to access the first application using interprocess communication;

identifying a second application that is attempting to access the first application using interprocess communication; and

rerouting the attempt to access the first application through an interprocess communication controller that determines whether to allow the attempt, based on rules indicating whether the second application may access the first application using interprocess communication.

26. The method of claim 25, wherein said registering step includes supplying rules specifying particular communications from which the first application is to be protected.

27. The method of claim 26, wherein the interprocess communication controller determines whether to allow the attempt based, at least in part, upon the rules specifying particular communications from which the first application is to be protected.

28. The method of claim 25, wherein said detecting step includes intercepting operating system calls for accessing the first application.

29. The method of claim 25, wherein said detecting step includes detecting a graphical device interface (GDI) message sent to the first application.

30. The method of claim 29, wherein said identifying step includes evaluating parameters of the message sent to the first application.

31. The method of claim 25, wherein said detecting step includes detecting an attempt to send keystoke data to a window of the first application.

32. The method of claim 25, wherein said detecting step includes detecting an attempt to send mouse movement data to a window of the first application.

33. The method of claim 25, wherein said rerouting step includes rerouting the

attempt to access the first application from a system dispatch table to the interprocess communication controller.

34. The method of claim 25, wherein said rules indicating whether the second application may access the first application includes rules indicating particular types of communications which are allowed.

35. The method of claim 25, further comprising:

if the interprocess communication controller allows the attempt to access the first application, routing the attempt to the first application.

36. A system for regulating interprocess communication between applications, the system comprising:

a computer having at least one processor, said computer operating under control of an operating system providing interprocess communication;

a policy specifying applications that are permitted to communicate with a first application using interprocess communication;

a module for detecting a second application attempting to communicate with the first application using interprocess communication; and

an interprocess communication controller for identifying the second application attempting to communicate with the first application and determining whether to permit the communication based upon the identification of the second application and the policy specifying applications permitted to communicate with the first application.

37. The system of claim 36, wherein said policy includes rules indicating particular types of communications which are permitted.

38. The system of claim 36, further comprising:

a rules engine for specifying applications that are permitted to communicate with the first application using interprocess communication.

39. The system of claim 36, further comprising:
a registration module for establishing said policy.

40. The system of claim 39, wherein said registration module provides for identifying applications to be governed by said policy.

41. The system of claim 36, wherein said module for detecting a second application detects an operating system call to open a communication channel to the first application.

42. The system of claim 36, wherein said module for detecting a second application detects a graphical device interface (GDI) message sent to the first application.

43. The system of claim 36, wherein said module for detecting a second application detects a local procedure call attempting to access the first application.

44. The system of claim 36, wherein said module for detecting a second application redirects attempts to communicate with the first application to the interprocess communication controller.

45. The system of claim 36, wherein said module for detecting a second application reroutes the attempt to communicate with the first application from a dispatch table to the interprocess communication controller.

46. The system of claim 36, wherein said interprocess communication controller determines whether to permit the communication based, at least in part, upon evaluating parameters of the attempt made by the second application to communicate with the first application.

47. The system of claim 36, wherein said interprocess communication controller

determines whether to permit the communication based upon obtaining user input as to whether to permit the second application to communicate with the first application.

8. EVIDENCE APPENDIX

This Appeal Brief is not accompanied by an evidence submission under §§ 1.130, 1.131, or 1.132.

9. RELATED PROCEEDINGS APPENDIX

Pursuant to Appellant's statement under Section 2, this Appeal Brief is not accompanied by any copies of decisions.